DynInt：面向大规模点击率预测的动态交互建模

Yachen Yan*Liubo Li*
yachen.yan@creditkarma.com
liubo.li@creditkarma.com
Credit Karma 美国加利福尼亚州
旧金山

arXiv:2301.08139v1 [cs.IR] 3 Jan 2023

摘要
学习特征交互是广告排序和推荐系统中大规模点击率预测取得成功的关键。在工业界，基于深度神经网络的模型已被广泛用于解决这类问题。研究人员提出了各种神经网络架构，以端到端的方式搜索和建模特征交互。然而，大多数方法只学习静态特征交互，尚未充分发挥深度点击率模型的表示能力。本文提出一种新模型：DynInt。DynInt 扩展了多项式交互网络（PIN），使其递归学习的高阶交互变为动态且依赖于数据；在此基础上，DynInt 进一步提出了两种建模动态高阶交互的模式：动态激活和动态参数。在动态激活模式下，我们通过实例感知激活门控网络，自适应地调整已学习交互的强度。在动态参数模式下，我们采用不同形式重新参数化参数，并通过实例感知参数生成网络动态生成参数。借助实例感知门控机制和动态参数生成，我们使 PIN 能够对潜在的工业应用进行动态交互建模。我们实现了所提出的模型，并在真实世界数据集上评估其性能。大量实验结果表明，与最先进的模型相比，DynInt 兼具高效性和有效性。

CCS 概念
• **计算方法**；• **机器学习**；• **机器学习方法**；• **神经网络**；

关键词
点击率预测、推荐系统、特征交互、深度神经网络

*两位作者对本研究贡献相同。

允许免费为个人或课堂使用而制作本作品的全部或部分数字副本或纸质副本，前提是这些副本不得用于或分发以获取利润或商业利益，并且副本须附有本声明及首页上的完整引文。对于本作品中由 ACM 以外的其他方拥有版权的内容，必须尊重其版权。注明出处后可以摘录。若要以其他方式复制、再版、发布到服务器或分发到列表，则须事先获得特别许可和/或支付费用。请向 permissions@acm.org 申请许可。*DLP-KDD 2022，2022年8月14日*，美国华盛顿特区© 2022 计算机协会。ACM ISBN 978-1-4503-XXXX-X/18/06...$15.00 https://doi.org/XXXXXXX.XXXXXXX

ACM 引用格式：
Yachen Yan 和 Liubo Li。2022。DynInt：大规模点击率预测的动态交互建模。载于华盛顿特区 *2022*：*ACM SIGKDD* 知识发现与数据挖掘会议，*2022* 年 *8* 月 *14–18* 日，华盛顿特区。ACM，纽约，纽约州，美国，9 页。https: //doi.org/XXXXXXX.XXXXXXX

1 引言
点击率（CTR）预测模型 [22] 是大规模搜索排序、在线广告和推荐系统 [4, 9, 18, 33]的重要组成部分。

业界已提出许多基于深度学习的模型来解决点击率预测问题，并且这些模型已成为以端到端方式学习混合类型输入中有用特征交互的主流方法[33]。尽管现有方法大多侧重于自动建模静态的深层特征表示，但动态建模特征表示的研究却很少。

然而，静态特征交互学习方法可能无法充分应对大规模搜索排序、在线广告和推荐系统问题的复杂性及其长尾属性。静态特征交互学习无法有效且高效地捕捉长尾实例的特征，因为短尾实例和高频特征很容易主导模型共享的静态参数。因此，我们认为，学习到的特征交互应具有动态性，并能针对不同实例进行自适应，以捕捉长尾实例的模式。

受动态模型框架启发，我们提出了一种名为 DynInt 的动态特征交互学习方法。DynInt 包含两种方案，可增强基础模型（xDeepInt [32]），以学习动态且个性化的交互。

• DynInt-DA：我们通过实例感知激活门控网络增强多项式交互网络，以自适应地细化不同 s 的已学习交互。

• DynInt-DGP 和 DynInt-DWP：我们通过实例感知参数生成网络对多项式交互网络的参数进行重新参数化，以自适应地生成参数并重新加权，从而学习动态交互。

• 我们引入不同的计算范式，以有效降低内存开销并提高实现效率。

• 我们引入正交正则化，以增强动态交互建模场景中所学习表示的多样性。

• 我们在真实世界数据集上进行了大量实验，并研究了不同超参数的影响，包括但不限于：核大小、潜在秩和正交正则化率。

2 相关工作

2.1 静态交互建模

大多数深度CTR模型会在初始步骤中，将高维稀疏类别特征和连续数值特征映射到低维潜在空间。许多现有模型架构侧重于同时学习静态的隐式/显式特征交互。

各种混合网络架构 [4, 7, 15, 19, 20, 27, 28]都采用带有非线性激活函数的前馈神经网络作为其深层组件，以学习隐式交互。对隐式交互建模的补充，提升了仅对显式交互进行建模的网络的性能 [2]。然而，这类方法无法限制所学习交互的阶数。

深度交叉网络（DCN） [27] 及其改进版本 DCN V2 [28] 以递归方式显式探索位级特征交互。深度因子分解机（DeepFM） [7]利用因子分解机层对成对的向量级交互进行建模。乘积神经网络（PNN） [19, 20] 引入内积层和外积层，分别学习向量级交互和位级交互。xDeepFM [15] 利用压缩交互网络（CIN）学习显式的向量级交互；CIN 具有类RNN架构，并使用哈达玛积学习向量级交互。FiBiNET [12] 利用挤压与激励网络动态学习特征的重要性，并通过双线性函数对特征交互进行建模。AutoInt [24] 借助 Transformer [26] 架构，学习输入特征的不同阶组合。xDeepInt [32] 利用多项式交互层，以受控的阶数递归地联合学习高阶向量级和位级交互，无需联合训练的深度神经网络和非线性激活函数。

2.2 深度学习中的门控机制

门控机制广泛应用于计算机视觉和自然语言处理领域，更重要的是，也应用于广告排序和推荐系统中的点击率预测。

LHUC [25] 学习用于说话人自适应的隐藏单元贡献。挤压与激励网络 [10] 通过显式建模通道间的依赖关系，并将每个通道与学习得到的门控值相乘，重新校准各通道的特征响应。门控线性单元（GLU） [6] 用于控制语言建模中的信息流带宽。

多门控专家混合（MMoE） [16]利用门控网络为每项任务自动分配共享专家表示的权重，从而同时建模共享信息和任务特定信息。GateNet [11] 利用特征嵌入门控和隐藏门控，分别从特征级和中间表示级中选择潜在信息。MaskNet [29] 引入 MaskBlock，在特征嵌入和中间输出上应用实例引导掩码。MaskBlock 可以串联或并联组合。个性化冷启动模块（POSO） [5] 引入面向用户群体的专用子模块，以增强现有模块的个性化能力，从而解决用户冷启动问题。具体而言，POSO 将个性化门控机制与多层感知机、多头注意力和多门控专家混合模型等现有模块相结合，并有效提升了它们的性能。

2.3 深度学习中的参数生成

除了调整特征嵌入和中间表示外，动态参数生成是使神经网络适应输入实例更直接的方式。

动态滤波器网络（DFN） [13] 和超网络 [8]分别是针对卷积神经网络和循环神经网络进行运行时参数生成的两种经典方法。DynamicConv [30], 则更简单高效，且已被证明其性能可与自注意力相媲美。

在推荐领域，PTUPCDR [35] 利用由用户特征嵌入输入的元网络生成个性化桥接函数，从而实现跨域推荐中的个性化迁移。协同行为网络（CAN） [3] 将可学习的物品嵌入用作微型多层感知机的参数，以实现与用户历史行为序列的交互，在不引入过多额外参数的情况下近似显式且独立的成对特征交互。自适应参数生成网络（APG） [31]通过低秩近似对深度点击率模型的全连接层参数重新参数化，并根据不同实例动态生成特定参数。

3 提出的模型：动态交互网络（DynInt）

本节概述 DynInt 变体的架构。我们首先说明符号。 F 表示特征数量。i 表示实例索引。 j 表示特征索引。 D 表示特征嵌入维度。 K 表示近似秩。 B 表示批大小。X_0 表示输入特征图，X_l 表示第 l 层的输出。 W_l 表示第 l 层的参数矩阵。

我们的模型首先将每个特征嵌入为一个维度为 D 的稠密嵌入向量x_j。我们将嵌入层$X_0 \in R^{F\times D}$ 的输出记为输入特征图：

$$X_0 = [x_0, x_1, \cdots, x_{F-1}]^\intercal. \tag{1}$$

所有 DynInt 模型变体的共享组件均来自 xDeepInt [32]。如图2所示，xDeepInt 的关键组件包括多项式交互网络（PIN）层、子空间交叉机制，以及 GroupLasso-FTRL/FTRL 复合优化策略：

• PIN 层能够捕获高阶多项式交互，是我们架构的骨干网络。

• 子空间交叉机制使 PIN 能够学习向量级交互和位级交互，同时控制向量级交互与位级交互之间的混合程度。如图2b所示，通过子空间交叉

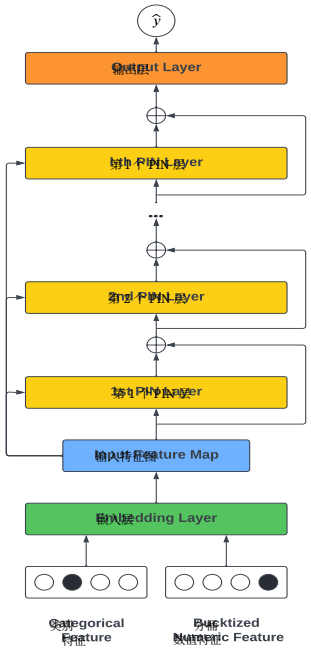
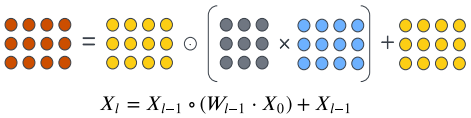
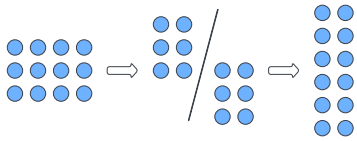


图1：带有残差连接的展开式多项式交互网络架构

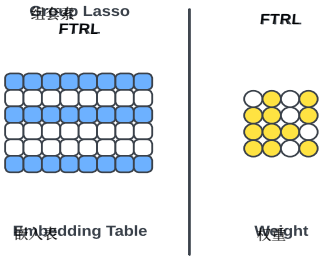


$$X_l = X_{l-1} \circ (W_{l-1} \cdot X_0) + X_{l-1}$$

(a) 多项式交互网络（PIN）层



(b) 子空间交叉机制



(c) GroupLasso FTRL 与 FTRL

图2：xDeepInt 模型的关键组件

机制，可以选择完整保留字段级语义信息，只探索特征之间的向量级交互。也可以将嵌入空间拆分为 h 子空间，以此建模位级交互。

• GroupLasso-FTRL/FTRL 复合优化策略利用了不同优化器的特性：GroupLasso-FTRL 为嵌入表实现行级稀疏性，而 FTRL 为权重参数实现元素级稀疏性，如图2c所示。

DynInt 模型变体的输出层只是一个作用于特征维度的全连接层。PIN 的输出是一个由不同阶数的特征交互构成的特征图，其中包括通过残差连接保留的输入特征图，以及由 PIN 学习到的高阶特征交互。输出层的作用如下：

$$\hat{y} = \sigma\left((W_{out} X_l + b_{out} 1^T)1\right) \tag{2}$$

其中，σ 为 Sigmoid 函数，$W_{out} \in R^{1\times F}$ 为聚合向量，用于线性组合所有学习到的特征交互，$b \in R$ 为偏置。

我们将主要致力于增强 PIN 层的架构，使模型具备实例感知能力。因此，我们将重点介绍 PIN 层的改进。为简化符号表示，模型介绍中不包含子空间交叉机制，但在实现中会将其纳入，以提升模型性能。

3.1 多项式交互网络

我们首先介绍 PIN 层的架构。带有残差连接的第 l个 PIN 层的数学表示如下：

$$X_l = X_{l-1} \circ (W_{l-1} \cdot X_0) + X_{l-1} \tag{3}$$

如图2a所示，X_l 是 PIN 的第 l层输出。$X_l \in R^{F\times D}$，其中 F是特征数量，D是嵌入维度。$W_l \in R^{F\times F}$是一个在不同实例间共享的静态参数。

为了增强 PIN 的表征能力，我们使 PIN 层参数具备实例感知能力，从而实现动态的向量级和位级交互建模。我们提出以下两种方案。

3.2 动态激活（DynInt-DA）

第一种方案是在每个 PIN 层的输出上添加动态机制。

$$X_l = \left(X_{l-1} \circ (W_{l-1} \cdot X_0)\right) \circ G_{l-1}^{(i)} + X_{l-1} \tag{4}$$

我们将第 l层的输出与一个实例感知的门控向量 $G(i)\,l$相乘。门控向量 $G(i)\,l$ 可以通过一个压缩比为 r 的双层深度神经网络建模，并以 X_0 作为输入。

通过实验，我们发现将 $G(i)\,l$的初始值设在 1.0 附近能够带来更好的收敛效果。我们将门控网络的激活函数设为：Sigmoid(x) $*$ 2.0。通过使用该激活

函数，当 DNN 层参数采用零均值初始化时，我们可以使 $G^{(i)}_l$ 的初始值接近 1.0。同时，我们使用输入特征图 X_0 作为门控网络的输入，以建模每个 PIN 层输出的动态门控。

在训练阶段，输入特征图 X_0 不会接收来自门控网络的梯度，以确保训练稳定性。图 3 展示了 DynInt-DA 模型增强型 PIN 层的架构。

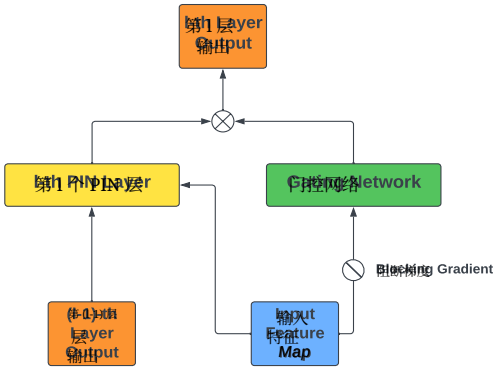


图 3：具有动态激活的 PIN 层。

3.3 动态参数（DynInt-D P)

动态参数方案旨在对每个 PIN 层进行动态建模 W_{l-1}。我们提出了两种动态参数建模方法。

3.3.1 动态生成参数（*DynInt-DGP*）。我们不再使用静态参数，而是允许参数矩阵依赖于数据。

$$X_l = X_{l-1} \circ (W^{(i)}_{l-1} \cdot X_0) + X_{l-1} \tag{5}$$

$W_{(i)\,l-1}$ 是第 i 个实例的动态参数矩阵。

3.3.2 动态加权参数（*DynInt-DWP*）。令 $G_{(i)\,l-1}$ 为第 i 个实例的动态加权矩阵。我们用动态加权矩阵 $G_{(i)\,l-1}$ 与静态参数矩阵 W_{l-1} 的哈达玛积替换静态参数矩阵 W_{l-1}，从而得到

$$X_l = X_{l-1} \circ \left((G^{(i)}_{l-1} \circ W_{l-1} \cdot X_0)\right) + X_{l-1} \tag{6}$$

在 DGP 和 DWP 中，动态参数矩阵 $W_{(i)\,l-1}$ 和动态加权矩阵 G^b_{l-1} 也可以通过压缩比为 r 的两层深度神经网络进行估计。考虑批大小后，反向传播需要计算大小为 $B \times F \times F$ 的矩阵的梯度，导致内存开销极高。直接实现 DWP 和 DGP 在实践中并不可行。在下一节中，我们提出两种计算范式，以较低的内存开销实现 DGP 和 DWP。

3.4 动态参数的低秩近似

低秩近似常用于降低内存和计算开销。在 PIN 层中，我们研究了训练后参数矩阵 W_l 的奇异值。如图 5 所示，奇异值急剧下降。

这一现象表明，大部分信息集中在最大的奇异值和对应的向量中。因此，对参数进行低秩近似能够有效地重构参数，同时提高服务效率。

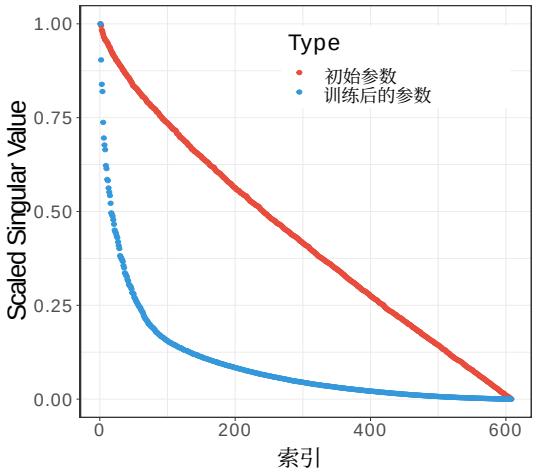


图 4：训练所得参数矩阵 W_l 的奇异值。我们观察到，训练所得静态参数矩阵的缩放奇异值呈现肘形，这表明对参数矩阵进行低秩近似能够保留其中的大部分信息。

3.5 动态参数公式的计算范式

计算范式假设动态参数矩阵或加权矩阵的秩较低，从而控制模型复杂度并简化计算。基于这一假设，我们对这些动态矩阵应用奇异值分解（SVD）。

3.5.1 动态生成参数。假设我们有动态参数矩阵的秩为 K 的近似 $W_{(i)\,l-1}$

$$W^{(i)}_{l-1} = U_{l-1} \cdot \Sigma^{(i)}_{l-1} \cdot V^T_{l-1} \tag{7}$$

其中 $U_{l-1} \in R^{F \times K}$, $\Sigma_{(i)\,l-1} \in R^{K \times K}$ 且 $V_{l-1} \in R^{F \times K}$。我们令奇异值矩阵随实例自适应。因此，所有实例共享由 U_{l-1} 和 V_{l-1} 张成的同一向量空间。这可以视为一种正则化技巧。于是，式（5）中的 $W_{(i)\,l-1} \cdot X_0$ 可以写为：

$$U_{l-1} \cdot \Sigma^{(i)}_{l-1} \cdot V^T_{l-1} \cdot X_0 \tag{8}$$

由于 U_{l-1} 和 V_{l-1} 是所有实例共享的静态参数，因此我们只需存储两个大小为 $F \times K$ 和 B 的对角

矩阵，并在前向传播期间存储两个大小为 $K \times K$ 的矩阵。由于 $\Sigma^{(i)}_{l-1}$ 取决于数据，我们将使用压缩比为 r 的双层深度神经网络，以 X_0 作为输入，对 $\Sigma^{(i)}_{l-1}$ 的非零对角元素进行建模。与 DynInt-DA 类似，我们阻断了 X_0 的梯度，以防止共适应问题。

图5a 展示了我们如何将 W_{l-1} 分解为静态参数 U_{l-1} 和 V_{l-1}，以及实例感知动态参数 $\Sigma^{(i)}_{l-1}$。

3.5.2 动态加权参数。假设 $G^{(i)}_{l-1}$ 是第（$l-1$）层的加权矩阵，其秩为 K。那么 $G^{(i)}_{l-1}$ 的奇异值分解形式为

$$G^{(i)}_{l-1} = \sum_{p=1}^{K} \tilde{u}^{(i)}_{p,l-1} \tilde{v}^{(i)}_{p,l-1} \tag{9}$$

其中 $\tilde{u}^{(i)}_{p,l-1} = \sqrt{\sigma^{(i)}_{p,l-1}} u^{(i)}_{p,l-1}$ 和 $\tilde{v}^{(i)}_{p,l-1} = \sqrt{\sigma^{(i)}_{p,l-1}} v^{(i)}_{p,l-1}$。$\tilde{u}^{(i)}_{p,l-1}$ 和 $\tilde{v}^{(i)}_{p,l-1}$ 是单位范数向量，并满足对于所有 p，$\tilde{u}^{(i)}_{p,l-1} \perp \tilde{v}^{(i)}_{p,l-1}$。因此，具有动态参数的 PIN 层可以重写为

$$X_l = \sum_{p=1}^{K} X_{l-1} \circ \left((\mathbf{diag}(\tilde{u}^{(i)}_{p,l-1}) \cdot W_{l-1} \cdot \mathbf{diag}(\tilde{v}^{(i)}_{p,l-1})) \cdot X_0 \right) + X_{l-1} \tag{10}$$

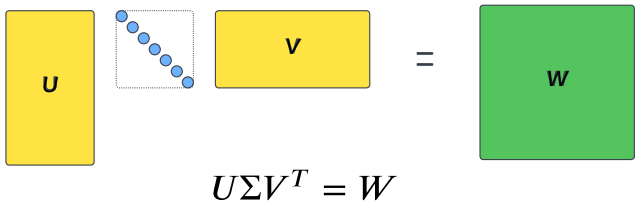
其中，$\tilde{u}^{(i)}_{p,l-1} \perp \tilde{v}^{(i)}_{p,l-1}$ 适用于所有 p。我们将哈达玛积简化为矩阵乘积。在这种形式下，W_{l-1} 是共享参数矩阵。$\tilde{u}^{(i)}_{p,l-1}$ 和 $\tilde{v}^{(i)}_{p,l-1}$ 是个性化权重。由于我们可以通过逐行广播操作和逐列广播操作，简化秩为 1 的矩阵与矩阵之间的哈达玛积，因此我们将向量化形式应用于 DWP。在每次前向传播迭代中，都会存储 K 对 $\tilde{u}^{(i)}_{p,l-1}$, $\tilde{v}^{(i)}_{p,l-1}$ 以及共享参数矩阵 W_{l-1}。

图5b展示了动态加权矩阵 $G^{(i)}_{l-1}$ 的分解。K 对 $\tilde{u}^{(i)}_{p,l-1}$, $\tilde{v}^{(i)}_{p,l-1}$ 由一个压缩比为 r 的双层深度神经网络建模，并以梯度阻断后的 X_0 作为输入，设置与前述方法类似。为了确保 $G^{(i)}_{l-1}$ 中的元素在初始化后接近 1.0，我们将 $1.0/K$ 加到用于建模 K 对 $\tilde{u}^{(i)}_{p,l-1}$, $\tilde{v}^{(i)}_{p,l-1}$ 的深度神经网络输出上。我们发现，这一技巧提升了训练稳定性。

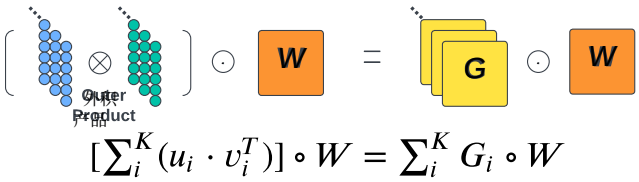
3.5.3 内存开销降低。对于 DynInt-DWP 和 DynInt-DGP，实例感知参数矩阵的内存开销为 $B \times F \times F$。对于采用所提出计算范式的 DynInt-DGP，参数矩阵低秩近似的内存开销为 $B \times K \times K + 2 \times F \times K$，其中 $K << F$。如果 $K = \frac{1}{10}F$，采用所提出的计算范式后，存储个性化加权矩阵的内存开销可降低 99%。对于采用所提出计算范式的 DynInt-DWP，

参数矩阵低秩近似的内存开销为 $2 \times B \times F \times K + F \times F$，其中 $K << F$。如果 $K = \frac{1}{10}F$，存储个性化加权矩阵的内存开销可降低 80%。

3.5.4 浮点运算次数分析。DynInt-DGP 对每个实例需要 $O(F \times F \times D)$ 浮点运算次数。通过奇异值分解，计算开销可降至 $O(2 \times F \times K \times K + F \times K \times D)$，奇异值分解的浮点运算次数显著



(a) 动态参数矩阵的低秩近似 W。在实现中，我们将 U 和 V 设为静态参数。Σ 是动态参数。



(b) 动态加权矩阵的低秩近似 G。在实现中，我们令 u_i 和 v_i 为动态参数。

图 5：两种动态参数公式的低秩近似。*DGP* 和 *DWP* 均使用奇异值分解来近似动态参数矩阵（DGP）或动态加权矩阵（DWP）。

在给定 $K << D$，F 时，小于 $O(F \times F \times D)$。DynInt-DWP 对每个实例也需要 $O(F \times F \times D)$ 浮点运算次数。使用奇异值分解时，计算开销为 $O(2K \times F \times F + K \times F \times F \times D)$，当 $K << D$，F 时，这与 $O(F \times F \times D)$ 相当

因此，所提出的计算范式可以显著降低内存开销。采用所提出的计算范式，DGP 也可以降低计算开销。

3.6 正交正则化和损失函数

这两种计算范式均基于低秩近似和奇异值分解。在对实例感知动态参数进行建模的同时，每个实例也共享某些模型参数。这有助于控制模型复杂度。

正交性对于保证性能至关重要。我们认为有两个原因。首先，SVD 要求具备正交性。我们需要奇异向量彼此正交。其次，经过 SVD 分解的 DGP 和 DWP 可以视为由多个秩一 DGP 和 DWP 组成的集成。

正交约束保证了每个奇异向量所张成的列空间彼此不同。因此，我们将正交惩罚项添加到损失函数中，以鼓励模型学习多样化的细粒度表示，并取得更好的性能。

3.6.1 DynInt-DGP 损失。

$$Loss = \sum_{i=1}^{N} \left(y_i log(\hat{y}_i) + (1-y_i) log(1-\hat{y}_i)\right) + \lambda \sum_{i=1}^{N} \sum_{l=1}^{L} \sum_{p>q}^{K} \| \frac{\tilde{\mathbf{u}}_{p,l-1}^{T} \tilde{\mathbf{u}}_{q,l-1}}{\|\tilde{\mathbf{u}}_{p,l-1}\| \cdot \|\tilde{\mathbf{u}}_{q,l-1}\|} \|_1 + \| \frac{\tilde{\mathbf{v}}_{p,l-1}^{T} \tilde{\mathbf{v}}_{q,l-1}}{\|\tilde{\mathbf{v}}_{p,l-1}\| \cdot \|\tilde{\mathbf{v}}_{q,l-1}\|} \|_1 \quad (11)$$

对于 DGP，我们将不同 PIN 层中 l_1 范数的 $\tilde{\mathbf{u}} \cdot l$ 对和 $\tilde{\mathbf{v}} \cdot l$ 对的余弦相似度之和作为惩罚项。式（11）中的惩罚项确保了 $\tilde{\mathbf{u}} \cdot l$ 对与 $\tilde{\mathbf{v}} \cdot l$ 对之间的正交性。

3.6.2 DynInt-DWP 损失。

$$Loss = \sum_{i=1}^{N} \left(y_i log(\hat{y}_i) + (1-y_i) log(1-\hat{y}_i)\right) + \lambda \sum_{i=1}^{N} \sum_{l=1}^{L} \sum_{p>q}^{K} \| \frac{\tilde{\mathbf{u}}_{p,l-1}^{(i)} \tilde{\mathbf{u}}_{q,l-1}^{(i)}}{\|\tilde{\mathbf{u}}_{p,l-1}^{(i)}\| \cdot \|\tilde{\mathbf{u}}_{q,l-1}^{(i)}\|} \|_1 + \| \frac{\tilde{\mathbf{v}}_{p,l-1}^{(i)} \tilde{\mathbf{v}}_{q,l-1}^{(i)}}{\|\tilde{\mathbf{v}}_{p,l-1}^{(i)}\| \cdot \|\tilde{\mathbf{v}}_{q,l-1}^{(i)}\|} \|_1 \quad (12)$$

对于 DWP，我们将不同 PIN 层和实例中的 l_1 范数项加到$\mathbf{u}^{(i)} \cdot l$ 对和 $\tilde{\mathbf{v}}^{(i)} \cdot l$ 对的余弦相似度上。公式（12）中的惩罚项确保了 $\tilde{\mathbf{u}}^{(i)} \cdot l$对与 $\tilde{\mathbf{v}}^{(i)} \cdot l$ 对之间的正交性。

4 实验

本节重点评估我们提出的模型的有效性，并探讨以下研究问题：

- **Q1**：在点击率预测问题中，我们提出的 DynInt 模型变体与各个基线模型相比表现如何?
- **Q2**：与仅学习静态特征交互的各个基线模型相比，我们提出的 DynInt 模型变体表现如何?
- **Q3**：不同的超参数设置如何影响 DynInt 变体的性能?
- **Q4**：正交正则化如何提升 DynInt 的有效性?

4.1 实验设置

4.1.1 数据集。我们在三个广泛用于研究的公开真实世界数据集上评估所提出的模型。

1. Criteo。[1] Criteo数据集来自2014年的Kaggle竞赛。此后，Criteo AI Lab正式发布了该数据集，供学术研究使用。该数据集包含13个数值特征和26个类别特征。我们通过变换函数 $\lfloor Log\left(V^2\right) \rfloor$ 将所有数值特征离散化为整数，并将其视为类别特征；这一做法由Criteo竞赛的获胜团队采用。

2. Avazu。[2] Avazu 数据集来自 2015 年的 Kaggle 竞赛。Avazu 提供了 10 天的点击数据。我们使用 21 个特征进行建模。该数据集中的所有特征都是类别特征。

3.iPinYou。[3] iPinYou数据集来自2013年iPinYou全球实时竞价（RTB）算法竞赛。我们遵循 [34] 的数据处理步骤，并考虑全部16个类别特征。

对于所有数据集，我们将样本随机划分为三部分：70% 用于训练，10% 用于验证，20% 用于测试。我们还会移除每个类别特征中出现次数少于 20 次的低频取值，以缓解稀疏问题。请注意，我们希望比较自动学习高阶特征交互的效果和效率，因此不进行任何特征工程，只进行特征转换，例如数值特征分桶和类别特征频率阈值筛选。

4.1.2 评估指标。我们使用 AUC 和对数损失来评估模型的性能。

对数损失对数损失既是我们的损失函数，也是评估指标。它衡量所有样本的预测概率与真实标签之间的平均距离。

AUCROC 曲线下面积（AUC）衡量模型将随机选取的正样本排在随机选取的负样本之前的概率。AUC 仅考虑正样本与负样本之间的相对顺序。AUC 越高，表示排序性能越好。

4.1.3 对比模型。我们将所有 DynInt 变体与以下模型进行比较：LR（逻辑回归） [17, 18], FM（因子分解机） [21], DNN（多层感知机）、Wide & Deep [4], DeepCrossing [23], DCN（深度交叉网络） [27],DCN V2 [28],PNN（同时包含内积层和外积层） [19, 20], DeepFM [7], xDeepFM [15], AutoInt [24] 以及 FiBiNET [12]。其中一些模型是点击率预测问题中最先进的模型，并广泛应用于工业界。

4.1.4 可复现性。我们使用 TensorFlow [1]实现所有模型。所有特征的嵌入维度均为 16，小批量大小为 4096。对于优化，所有神经网络模型均采用 Adam [14]，学习率在 10^{-4} 至 10^{-3} 之间调优；LR 和 FM 则采用 FTRL [17, 18]，学习率在 10^{-2} 至 10^{-1} 之间调优。对于正则化，我们为全连接层采用 L2 正则化，其 λ 取值范围为 10^{-4} 至 10^{-3}。我们在验证数据集上对每个对比模型的超参数进行网格搜索。全连接层或交互层的数量为 1 至 4 层，神经元数量范围为 128 至 1024。所有模型均采用早停训练，并每隔 2000 个训练步进行评估。

与 [32], 类似，xDeepInt 和 DynInt 的超参数搜索设置如下：递归特征交互层的数量在 1 到 4 之间搜索。对于子空间的数量 h，搜索值为 1、2、4、8 和 16。由于嵌入维度为 16，这一范围涵盖了从完全向量级交互到完全位级交互的所有情况。对于所有 DynInt 变体的压缩比，我们在 2 到 16 之间搜索。
DynInt-DGP 的核大小在 16 到 128 之间调整。DynInt-DWP 加权矩阵的潜在秩在 1 到 4 之间搜索。我们使用 G-FTRL

[1]https://www.kaggle.com/c/criteo-display-ad-challenge [2] https://www.kaggle.com/c/avazu-ctr-prediction

[3]http://contest.ipinyou.com/

优化器优化嵌入表，并使用 FTRL 优化动态 PIN 层，学习率在 10^{-2} 到 10^{-1} 之间调整。

4.2 模型性能比较（Q1）

表1：不同算法在 Criteo、Avazu 和 iPinYou 数据集上的性能比较。

	Criteo		Avazu		iPinYou	
模型	AUC	对数损失	AUC	对数损失	AUC	对数损失
LR	0.7924	0.4577	0.7533	0.3952	0.7692	0.005605
FM	0.8030	0.4487	0.7652	0.3889	0.7737	0.005576
DNN	0.8051	0.4461	0.7627	0.3895	0.7732	0.005749
Wide&Deep	0.8062	0.4451	0.7637	0.3889	0.7763	0.005589
DeepFM	0.8069	0.4445	0.7665	0.3879	0.7749	0.005609
DeepCrossing	0.8068	0.4456	0.7628	0.3891	0.7706	0.005657
DCN	0.8056	0.4457	0.7661	0.3880	0.7758	0.005682
PNN	0.8083	0.4433	0.7663	0.3882	0.7783	0.005584
xDeepFM	0.8077	0.4439	0.7668	0.3878	0.7772	0.005664
AutoInt	0.8053	0.4462	0.7650	0.3883	0.7732	0.005758
FiBiNET	0.8082	0.4439	0.7652	0.3886	0.7756	0.005679
DCN V2	0.8086	0.4433	0.7662	0.3882	0.7765	0.005593
xDeepInt	0.8111	0.4408	0.7672	0.3876	0.7790	0.005567
DynInt-DA	0.8125	0.4398	0.7677	0.3871	0.7802	0.005552
DynInt-DGP	0.8127	0.4397	**0.7686**	**0.3867**	**0.7813**	**0.005547**
DynInt-DWP	**0.8132**	**0.4393**	0.7671	0.3878	0.7798	0.005564

不同模型架构的整体性能列于表1中。关于模型有效性，我们有以下观察：

• 与LR基线模型相比，随着模型复杂度的增加，因子分解机（FM）带来了最显著的相对性能提升。这表明学习显式的向量级特征交互非常重要。

• 同时学习向量级和位级交互的模型始终优于其他模型。这一现象表明，两种特征交互对于预测性能都至关重要，并且能够相互补足。

• 在所有静态交互学习模型中，xDeepInt 的预测性能最佳。这一优势可能归因于：xDeepInt 通过调整 PIN 的深度来建模次数受限的多项式特征交互，并通过改变子空间数量来实现复杂度各异的位级特征交互。

• 与其他基线模型相比，xDeepInt 和所有 DynInt 模型变体即使不结合深度神经网络来学习特征嵌入中的隐式交互，也能取得更好的性能。这表明，在极端高维稀疏数据场景下，近似效果良好的多项式交互有望超越深度神经网络学得的隐式交互。

• 所有 DynInt 模型变体的表现都优于所有静态交互学习模型，这表明学习动态交互可以进一步提升模型性能。尽管不同数据集更适合不同类型的动态交互建模方法，但这主要是因为 DynInt 模型变体在设计上具有不同的模型容量和正则化效应。我们将在后续章节详细讨论这些模型。

4.3 特征交互层比较（Q2）我们在 Criteo 数据集上评估了每个基线模型，**不与深度神经网络集成**，也就是说，只应用特征交互层。不同特征交互层的整体表现列于表 2 中。我们可以观察到，最初与深度神经网络集成的大多数基线模型（DCN V2 除外）性能都显著下降。xDeepInt 和 DynInt 变体在学习隐式交互时没有与深度神经网络集成，因此保持了原有性能。基于以上观察，我们得出以下认识：

• 大多数基线模型都严重依赖深度神经网络来弥补特征交互层的不足，因此整体性能可能主要由深度神经网络部分决定。

• 对于 DeepFM 和 PNN 等模型架构，其交互层只能学习二阶交互，因此性能不如 DCN V2、xDeepInt 和 DynInt 变体。

• DCN V2、xDeepInt 和 DynInt 变体只能学习多项式交互，但性能仍优于其他基线模型，这表明数据集中确实存在高阶交互，且多项式交互对于取得优异性能至关重要。

表 2：Criteo数据集上不同特征交互层的性能比较。

	AUC	对数损失
DeepFM	0.8030	0.4487
DCN	0.7941	0.4566
PNN	0.8066	0.4451
xDeepFM	0.8065	0.4454
AutoInt	0.8056	0.4458
DCN V2	0.8082	0.4436
xDeepInt	0.8111	0.4408
DynInt-DA	0.8125	0.4398
DynInt-DGP	0.8127	0.4397
DynInt-DWP	0.8132	0.4393

4.4 超参数研究（Q3 和 Q4）

为了深入了解所提出的模型，我们在三个数据集上开展实验，并比较 DynInt 的多个变体在不同超参数设置下的表现。本节评估模型性能随以下超参数变化的情况：1）动态 PIN 层的深度；2）子空间数量；3）DynInt-DGP 生成参数的核大小；4）DynInt-DWP 加权矩阵的潜在秩；5）缩减率；6）DynInt-DGP 和 DynInt-DWP 的正交正则化率

4.4.1 网络深度。动态 PIN 层的深度决定了所学习的特征交互阶数。表3展示了模型性能随层数变化的情况。当层数设为 0 时，我们的模型等同于逻辑回归，不会学习任何交互。在此实验中，我们将子空间数量设为 1，以禁用按位特征交互。由于我们主要考察性能

相对于深度变化的情况，为简化起见，我们对 DynInt-DA、DynInt-DGP 和 DynInt-DWP 的性能取平均值。

Ta表 3：超参数的影响：层数 层

数据集	#层数	0	1	2	3	4	5
Criteo	AUC	0.7921	0.8058	0.8068	0.8075	**0.8085**	0.8081
	对数损失	0.4580	0.4455	0.4445	0.4440	**0.4432**	0.4436
Avazu	AUC	0.7547	0.7664	0.7675	**0.7682**	0.7678	0.7670
	对数损失	0.3985	0.3882	0.3870	**0.3868**	0.3872	0.3877
iPinYou	AUC	0.7690	0.7749	0.7787	**0.7797**	0.7785	0.7774
	对数损失	0.005604	0.005575	0.005569	**0.005565**	0.005578	0.005582

4.4.2 子空间数量。子空间交叉机制使所提出的模型能够控制位级交互的复杂度。表 4 展示了子空间交叉机制能够提升性能。在本实验中，我们将 PIN 层的数量设为 3。通常这是一个不错的选择，但不一定是每个数据集的最佳设置。由于我们主要考察性能随子空间数量变化的情况，为简便起见，我们对 DynInt-DA、DynInt-DGP 和 DynInt-DWP 的性能取平均值。

表 4：超参数的影响：子空间数量

数据集	#子空间	1	2	4	8	16
Criteo	AUC	0.8079	0.8085	0.8092	0.8107	**0.8125**
	对数损失	0.4439	0.4433	0.4421	0.4408	**0.4397**
Avazu	AUC	0.7670	0.7677	**0.7684**	0.7678	0.7672
	对数损失	0.3875	0.3872	**0.3867**	0.3872	0.3873
iPinYou	AUC	0.7782	0.7792	**0.7796**	0.7790	0.7784
	对数损失	0.005579	0.005568	**0.005565**	0.005568	0.005582

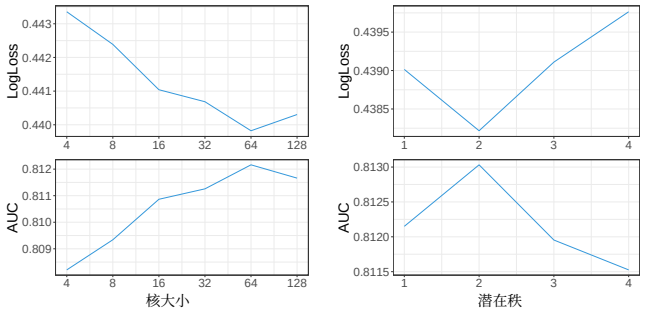
4.4.3 DynInt-DGP 的动态生成参数的核大小。DynInt-DGP 的核大小决定了 DynInt-DGP 的静态参数和动态参数的数量。

图6a展示了 DynInt-DGP 参数矩阵的核大小对其在 Criteo数据集上性能的影响。我们观察到，随着核大小增大，模型性能持续提升，直至核大小达到 64。这与我们对模型性能随模型复杂度变化的理解一致，也意味着核大小可作为正则化超参数，用于逼近底层真实权重。

4.4.4 DynInt-DWP 动态加权参数的潜在秩。DynInt-DWP 的潜在秩控制着共享静态参数动态加权矩阵的复杂度。

图6b展示了 DynInt-DGP 的参数矩阵潜在秩对其在 Criteo数据集上的性能的影响。其复杂度
增大加权矩阵的潜在秩时，DynInt-DWP 会快速增长。

4.4.5 压缩比。压缩比是一个超参数，可用于控制网络进行动态参数加权/生成的容量。我们在 Criteo数据集上考察了压缩比的影响。不同架构中，压缩比与性能之间的相关性相似。其他超参数均保持为各自架构的最佳设置。由于实验中各种方法呈现出相似的趋势，为了更直观地展示结果，我们对它们的性能取了平均值。



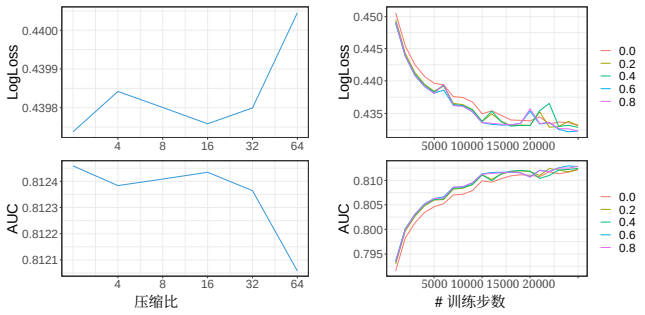
(a) DynInt-DGP 的核大小 **(b) DynInt-DWP 的秩**

图 6：对数损失和 AUC 随核大小和潜在秩的变化。

图 7a 展示了不同压缩比设置与性能之间的关系。我们发现，在合理范围内，性能对压缩比具有较强的稳健性。复杂度增加并不一定能提升性能，而较大的压缩比则能提高模型效率。

4.4.6 正交正则化率。正交正则化率是一个超参数，用于控制动态参数的列空间之间的表示多样性。由于动态生成/加权矩阵可视为低秩矩阵之和，因此每个分解子空间都应学习相对独特的特征。我们考察了正交正则化率对 Criteo数据集的影响。为简洁起见，我们对两种方法的性能取平均值，因为在实验中它们呈现出相似的趋势。

图 7b 展示了采用不同比例的正交正则化时，模型性能随训练步数的变化情况。我们观察到，正交正则化能够提升整体性能。我们还发现，采用相对较大的正交正则化时，整体性能大致相同，但较大的正交正则化仍能提高模型的收敛速度。这表明，增强潜在列空间的正交性能够有效提升动态参数生成/加权的质量。



(a) 压缩比 **(b) 正交正则化**

图 7：对数损失和 AUC 随压缩比及正交正则化率的变化。

5 结论

动态参数化已成为利用深度神经网络表示能力的一种流行方式。我们设计了网络 DynInt，并采用两种方案来增强模型容量和动态特征交互能力。借助基于低秩近似的计算范式，我们减少了实现中的内存开销。实验结果表明，在真实世界数据集上，无论是模型性能还是效率，我们的方法都优于最先进的算法。

在后续工作中，我们希望研究如何将我们的模型与其他架构有效、动态地结合起来，从而共同学习不同类型的交互。除了仅对定长特征进行建模外，我们还希望扩展我们的架构，使其能够针对现实问题中的序列行为学习变长特征。

参考文献

[1] Martín Abadi、Paul Barham、Jianmin Chen、Zhifeng Chen、Andy Davis、Jeffrey Dean、Matthieu Devin、Sanjay Ghemawat、Geoffrey Irving、Michael Isard 等。2016。Tensorflow：一个面向大规模机器学习的系统。发表于第*12*届 {*USENIX*} 操作系统设计与实现研讨会（{*OSDI*} *16*。265–283。[2] Alex Beutel、Paul Covington、Sagar Jain、Can Xu、Jia Li、Vince Gatto 和 Ed H Chi。2018。Latent Cross：利用循环推荐系统中的上下文。发表于第*11*届*ACM*国际网络搜索与数据挖掘会议论文集。ACM，46–54。[3] Weijie Bian、Kailun Wu、Lejian Ren、Qi Pi、Yujing Zhang、Can Xiao、Xiang-Rong Sheng、Yong-Nan Zhu、Zhangming Chan、Na Mou 等。2022。CAN：用于点击率预测的特征协同行为网络。发表于第*15*届*ACM*国际网络搜索与数据挖掘会议论文集。57–65。[4] Heng-Tze Cheng、Levent Koc、Jeremiah Harmsen、Tal Shaked、Tushar Chandra、Hrishi Aradhye、Glen Anderson、Greg Corrado、Wei Chai、Mustafa Ispir 等。2016。面向推荐系统的宽度与深度学习。发表于推荐系统深度学习研讨会论文集。ACM，7–10。[5] Shangfeng Dai、Haobin Lin、Zhichen Zhao、Jianying Lin、Honghuan Wu、Zhe Wang、Sen Yang 和 Ji Liu。2021。POSO：大规模推荐系统的个性化冷启动模块。*arXiv*预印本 *arXiv:2108.04690*（2021）。[6] Yann N Dauphin、Angela Fan、Michael Auli 和 David Grangier。2017。使用门控卷积网络进行语言建模。发表于国际机器学习会议。PMLR，933–941。[7] Huifeng Guo、Ruiming Tang、Yunming Ye、Zhenguo Li 和 Xiuqiang He。2017。DeepFM：用于点击率预测的因子分解机神经网络。*arXiv*预印本 *arXiv:1703.04247*（2017）。[8] David Ha、Andrew Dai 和 Quoc V Le。2016。超网络。*arXiv*预印本 *arXiv:1609.09106*（2016）。[9] Xinran He、Junfeng Pan、Ou Jin、Tianbing Xu、Bo Liu、Tao Xu、Yanxin Shi、Antoine Atallah、Ralf Herbrich、Stuart Bowers 等。2014。Facebook广告点击预测的实践经验。发表于第八届在线广告数据挖掘国际研讨会论文集。ACM，1–9。[10] Jie Hu、Li Shen 和 Gang Sun。2018。挤压与激励网络。发表于*IEEE*计算机视觉与模式识别会议论文集。7132–7141。[11] Tongwen Huang、Qingyun She、Zhiqiang Wang 和 Junlin Zhang。2020。GateNet：用于点击率预测的门控增强深度网络。*arXiv*预印本 *arXiv:2007.03519*（2020）。[12] Tongwen Huang、Zhiqi Zhang 和 Junlin Zhang。2019。FiBiNET：结合特征重要性与双线性特征交互的点击率预测模型。*arXiv*预印本 *arXiv:1905.09433*（2019）。[13] Xu Jia、Bert De Brabandere、Tinne Tuytelaars 和 Luc V Gool。2016。动态滤波器网络。神经信息处理系统进展 29（2016）。[14] Diederik P Kingma 和 Jimmy Ba。2014。Adam：一种随机优化方法。*arXiv*预印本 *arXiv:1412.6980*（2014）。[15] Jianxun Lian、Xiaohuan Zhou、Fuzheng Zhang、Zhongxia Chen、Xing Xie 和 Guangzhong Sun。2018。xdeepfm：结合显式与隐式特征交互的推荐系统模型。发表于第*24*届*ACM SIGKDD*国际知识发现与数据挖掘会议论文集。ACM，1754–1763。[16] Jiaqi Ma、Zhe Zhao、Xinyang Yi、Jilin Chen、Lichan Hong 和 Ed H Chi。2018。使用多门混合专家模型对多任务学习中的任务关系进行建模。发表于第*24*届*ACM SIGKDD*国际知识发现与数据挖掘会议论文集。1930–1939。[17] H Brendan McMahan。2011。跟随正则化领导者与镜像下降：等价定理和L1正则化。（2011）。[18] H Brendan McMahan、Gary Holt、David Sculley、Michael Young、Dietmar Ebner、Julian Grady、Lan Nie、Todd Phillips、Eugene Davydov、Daniel Golovin 等。2013。广告点击预测：一线实践观察。发表于第*19*届*ACM SIGKDD*国际知识发现与数据挖掘会议论文集。ACM，1222–1230。[19] Yanru Qu、Han Cai、Kan Ren、Weinan Zhang、Yong Yu、Ying Wen 和 Jun Wang。2016。用于用户响应预测的基于乘积的神经网络。发表于*2016*年*IEEE*第*16*届国际数据挖掘会议（*ICDM*）。IEEE，1149–1154。[20] Yanru Qu、Bohui Fang、Weinan Zhang、Ruiming Tang、Minzhe Niu、Huifeng Guo、Yong Yu 和 Xiuqiang He。2018。用于多字段类别数据用户响应预测的基于乘积神经网络。*ACM*信息系统汇刊（*TOIS*）37卷1期（2018），5。[21] Steffen Rendle。2010。因子分解机。发表于*2010*年*IEEE*国际数据挖掘会议。IEEE，995–1000。[22] Matthew Richardson、Ewa Dominowska 和 Robert Ragno。2007。点击预测：估算新广告的点击率。发表于第*16*届国际万维网会议论文集。ACM，521–530。[23] Ying Shan、T Ryan Hoens、Jian Jiao、Haijing Wang、Dong Yu 和 JC Mao。2016。深度交叉：无需人工设计组合特征的Web规模建模。发表于第*22*届*ACM SIGKDD*国际知识发现与数据挖掘会议论文集。ACM，255–262。[24] Weiping Song、Chence Shi、Zhiping Xiao、Zhijian Duan、Yewen Xu、Ming Zhang 和 Jian Tang。2018。AutoInt：通过自注意力神经网络实现自动特征交互学习。*arXiv*预印本 *arXiv:1810.11921*（2018）。[25] Pawel Swietojanski 和 Steve Renals。2014。学习隐藏单元贡献，实现神经网络声学模型的无监督说话人自适应。发表于*2014*年*IEEE*口语语言技术研讨会（*SLT*）。IEEE，171–176。[26] Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N Gomez、Łukasz Kaiser 和 Illia Polosukhin。2017。注意力机制即所需的一切。发表于神经信息处理系统进展。5998–6008。[27] Ruoxi Wang、Bin Fu、Gang Fu 和 Mingliang Wang。2017。用于广告点击预测的深度交叉网络。发表于*ADKDD'17*论文集。ACM，12。[28] Ruoxi Wang、Rakesh Shivanna、Derek Cheng、Sagar Jain、Dong Lin、Lichan Hong 和 Ed Chi。2021。DCN V2：改进的深度交叉网络及Web规模学习排序系统的实践经验。发表于*2021*年*Web*会议论文集。1785–1797。[29] Zhiqiang Wang、Qingyun She 和 Junlin Zhang。2021。MaskNet：通过实例引导掩码将逐特征乘法引入点击率排序模型。*arXiv*预印本 *arXiv:2102.07619*（2021）。[30] Felix Wu、Angela Fan、Alexei Baevski、Yann N Dauphin 和 Michael Auli。2019。使用轻量级动态卷积，减少注意力计算。*arXiv*预印本 *arXiv:1901.10430*（2019）。[31] Bencheng Yan、Pengjie Wang、Kai Zhang、Feng Li、Jian Xu 和 Bo Zheng。2022。APG：用于点击率预测的自适应参数生成网络。*arXiv*预印本 *arXiv:2203.16218*（2022）。[32] Yachen Yan 和 Liubo Li。2020。xDeepInt：用于建模向量级与位级特征交互的混合架构。（2020）。[33] Shuai Zhang、Lina Yao、Aixin Sun 和 Yi Tay。2019。基于深度学习的推荐系统：综述与新视角。*ACM*计算综述（*CSUR*）52卷1期（2019），5。[34] Weinan Zhang、Shuai Yuan、Jun Wang 和 Xuehua Shen。2014。基于亿玛数据集的实时竞价基准测试。*arXiv*预印本 *arXiv:1407.7073*（2014）。[35] Yongchun Zhu、Zhenwei Tang、Yudan Liu、Fuzhen Zhuang、Ruobing Xie、Xu Zhang、Leyu Lin 和 Qing He。2022。跨域推荐中的用户偏好个性化迁移。发表于第*15*届*ACM*国际网络搜索与数据挖掘会议论文集。1507–1515。